Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 24, 2013
H. Alvestrand
Google
October 21, 2012

Resolution Constraints in Web Real Time Communications
draft-alvestrand-constraints-resolution-01

Abstract

This document specifies the constraints necessary for a Javascript application to successfully indicate to a browser that supports WebRTC what resolutions it desires on a video stream.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

There are a number of scenarios where it's useful for a WebRTC application to indicate to the WebRTC implementation in the supported browser what the desired characteristics of a video stream are. These include, but are not limited to:

o Specifying a minimum desired resolution for a given application, in order to control the user experience or resource tradeoffs made by the browser to favour a particular stream

o Specifying a maximum desired resolution for a given stream, in order to save some resource (bandwidth, CPU....), possibly outside of the browser where the browser can't tell that it's exceeding a constraint

o Specifying resolutions that are a reasonable fit for the current usage of the video stream, for instance fitting with the number of pixels available on the part of a device's display surface that is devoted to displaying this video stream

o Specifying the shape of a video stream, in order to fit the video onto a display surface without the need for black bars or image distortion

Similar considerations apply for framerate.

1.1. Disposition of this text

This draft is written in order to get something specific out to refer to during spec-writing and implementation. The text may eventually get merged into the JSEP specification, [I-D.ietf-rtcweb-jsep].

2. Usage Scenarios

These constraints are usable in several places:

o As constraints to the getUserMedia call [W3C.WD-mediacapture-streams-20120628], where they serve to guide the configuration of the camera obtained, and may influence the choice of camera.

o As constraints to the addStream call on a PeerConnection [W3C.WD-webrtc-20120821], where they serve to guide the configuration of the codec that encodes the video content for transmission.

- As constraints applied to an existing local video stream using the "change constraints" API, where it may cause the video engine to reconfigure the device or codec for that particular stream.

- As constraints applied to an incoming video stream using the "change constrains" API on a MediaStreamTrack, where it serves to inform the video engine about the desirable properties of the video track, which may lead to the video engine choosing to reencode the video and/or signal a remote video source that it wishes certain constraints to be put in place.

All of the constraints may be meaningful in both "mandatory" and "optional" forms.

Consider the following (simplified) model of a video stream through a WebRTC application:

```
   |<-------------- Browser A -------------------->|
  Camera ---> Mediastream A ---> Peerconnection A ------+
          |<------- Application A ---------->|          |
                          v  ^                          v
                Signalling channel             Internet (media)
                          v  ^                          |
          |<------- Application B ---------->|          |
  <video> tag <-- Mediastream B <--- Peerconnection B --+
   |<-------------Browser B ---------------------->|
```

Both applications are running in browsers, with Application A connected to a camera that is able to deliver video streams up to HD quality (1280x720).

2.1. Scenario: Resolution change

At one particular moment in time, the <video> tag in Application B is rendered as a thumbnail, and video is flowing to it in a 160x100 resolution; there is no need to send any more data, since no more pixels are available for its display anyway.

Then the user of Application B hits the "full-screen" button. There are now 1600x1200 pixels available for display.

Initially, Application B will splay the 160x100 image across the larger surface, because there is no other choice, but it will desire to have as many pixels as possible available to provide a high quality image.

The choices available to the writer of application B are:

- o Signal (by non-standard means) to Application A that more pixels are needed. Application A will then modify the constraints on Mediastream A to say that the desired (not mandatory) min resolution is 1600x1200; Browser A will then reconfigure the camera to generate the closest available resolution, which is 1280x720.

- o Apply a new constraint set to Mediastream B's video track, saying that the desired resolution is now 1600x1200. Browser B will then have to figure out that this is an incoming track via Peerconnection B, and that the resolution needs to be signalled; it will then fire a NegotationNeeded event at Application B, which will then renegotiate the desired resolutions using an SDP exchange with Browser A; Browser A will then figure out from the SDP that it's useful to generate a higher resolution video stream, and reconfigure the camera as above.

- o Execute a renegotiation with Application A, adding a=remote-ssrc: attributes as described in Section 3 by modifying the SDP generated by CreateOffer, and triggering the behaviour in the previous alternative inside Browser A. API-wise, this is perhaps the most complex method.

The advantage of the first method is that it does not require any SDP parsing or generation.

The advantage of the second method is that it will work when appliation A and application B are different applications; there is no need for them to have any private agreement on how to set bitrate.

In the opinion of the author, there are no obvious advantages to the third method when the second method is available.

2.2. Scenario: Constrained bandwidth

At one particular moment in time, the camera is generating 1280x720, resulting in a 2 Mbits/second data flow from A to B. Congestion control signals that this data rate is no longer available; rather than letting the browser reduce the bandwidth of some flow of its choice, Application A decides that the high definition video is the feature that is least valuable. It can then apply a new constraint to Mediastream A, specifying that resolution should be at most 640x360; browser A is then responsible for making sure this decision is communicated to browser B (if it needs to be).

3. Syntax and Mapping Examples

See Section 4 for the actual definition of the constraints used here.

3.1. Examples with GetUserMedia

A constraint saying that we absolutely must have a minimum resolution of 1024x768:

```
getUserMedia({
   video: { mandatory: { minWidth: 1024, minHeight: 768 } }
}, successCallback, errorCallback);
```

A constraint saying that we'd prefer 60 frames per second, if available, and if we can get that, we'd like to limit the max resolution, but in all cases, the screen must be clamped to a 4:3 aspect ratio - 16:9 or odd aspect ratios are not acceptable to this application:

```
getUserMedia({
   video: {
     mandatory: { minAspectRatio: 1.333, maxAspectRatio: 1.334 },
     optional [
       { minFrameRate: 60 },
       { maxWidth: 640 },
       { maxHeigth: 480 }
     ]
   }
}, successCallback, errorCallback);
```

3.2. SDP mappings

This document does not specify or constrain how constraints get reflected into SDP (if they do); that's an implementor decision.

The examples below are thought exercises, based on [I-D.lennox-mmusic-sdp-source-selection] and [I-D.alvestrand-rtcweb-resolution].

An optional constraint has been applied to an incoming stream where both upper and lower are constrained to 320x200. The stream has been assigned to a hardware video decoder that can decode most resolutions up to 1024x768, in any aspect ratio, but only if all divisions are divisible by 4. The incoming stream has SSRC 1234.

Escaped line breaks are added for readability.

```
m=video
a=remote-ssrc:1234 imageattr:* [x=320,y=200,q=1.0] \
                  [x=[120:4:1024],y=[100:4:768],q=0.2]
```

4. IANA Considerations

This document requests IANA to register constraints in the "RTCWeb Media Constraints" registry created by [I-D.burnett-rtcweb-constraints-registry]. NOTE: The registrations assume that this document is updated to no longer have "video" as part of the name, but have "video" as a field-of-use in the registration.

The definitions of width, height and aspect ratio are taken from [RFC6236].

- o minWidth - valid for video. Corresponds to the "x" value (pixel count) from RFC 6236. Only integer values are valid.
- o maxWidth - valid for video. Definition as for minWidth.
- o minHeight - valid for video. Corresponds to the "y" value (pixel count) from RFC 6236. Only integer values are valid.
- o maxHeight - valid for video. Definition as for minHeight.
- o minAspectRatio - valid for video. Corresponds to the "par" (picture aspect ratio), with "sar" set to 1.0. A 4:3 format display corresponds to an AspectRatio of 1.3333. Floating point values are valid.
- o maxAspectRatio - valid for video. Definition as for minAspectRatio.
- o minFramerate - valid for video. Corresponds to the framerate defined in [RFC4566], the "a=framerate" attribute.
- o maxFramerate - valid for video. Definition as for minFramerate.

The contact person is Harald Alvestrand <hta@google.com>.

Change control for the registration is with the IETF, as designated by the IESG.

Note that minFramerate defines a lower bound for the a=framerate attribute, which is itself defined as an upper limit; this means that even if a high framerate is negotiated, the actual framerate used may be lower due to temporary considerations (for instance CPU or bandwidth, or simply lack of movement in the picture).

5. Security Considerations

No security considerations particular to these specific constraints have so far been identified.

6. Acknowledgements

Special thanks are given to Dan Burnett, Cullen Jennings, the IETF RTCWEB WG and the W3C WEBRTC WG for strongly influencing this memo, and to Per Kjellander for being the first to implement the constraints in getUserMedia.

7. References

7.1. Normative References

[I-D.burnett-rtcweb-constraints-registry]
Burnett, D., "IANA Registry for RTCWeb Media Constraints", draft-burnett-rtcweb-constraints-registry-01 (work in progress), April 2012.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[RFC6236] Johansson, I. and K. Jung, "Negotiation of Generic Image Attributes in the Session Description Protocol (SDP)", RFC 6236, May 2011.

7.2. Informative References

[I-D.alvestrand-rtcweb-resolution]
Alvestrand, H., "RTCWEB Resolution Negotiation", draft-alvestrand-rtcweb-resolution-00 (work in progress), April 2012.

[I-D.ietf-rtcweb-jsep]

Uberti, J. and C. Jennings, "Javascript Session Establishment Protocol", draft-ietf-rtcweb-jsep-01 (work in progress), June 2012.

[I-D.lennox-mmusic-sdp-source-selection]
Lennox, J. and H. Schulzrinne, "Mechanisms for Media Source Selection in the Session Description Protocol (SDP)", draft-lennox-mmusic-sdp-source-selection-04 (work in progress), March 2012.

[W3C.WD-mediacapture-streams-20120628]
Burnett, D. and A. Narayanan, "Media Capture and Streams", World Wide Web Consortium WD WD-mediacapture-streams-20120628, June 2012, <http://www.w3.org/TR/2012/WD-mediacapture-streams-20120628>.

[W3C.WD-webrtc-20120821]
Bergkvist, A., Burnett, D., Jennings, C., and A. Narayanan, "WebRTC 1.0: Real-time Communication Between Browsers", World Wide Web Consortium WD WD-webrtc-20120821, August 2012, <http://www.w3.org/TR/2012/WD-webrtc-20120821>.

Appendix A. Changes from -00 to -01

Added the "Usage Scenarios" chapter.

Repointed the eventual target to be incorporation in the JSEP draft.

Made sure the constraints are consistently spelled in camelCase, with a small initial letter.

Author's Address

Harald Alvestrand
Google

Email: harald@alvestrand.no